3

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/640,620	08/12/2003	Arra E. Avakian	10017134-1	1119

22879 7590 04/19/2007
HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
VU, TUAN A	
ART UNIT	PAPER NUMBER
2193	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	04/19/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No. 10/640,620	Applicant(s) AVAKIAN ET AL.
	Examiner Tuan A. Vu	Art Unit 2193

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *09 April 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 4/09/2007.

As indicated in Applicant's response, claim 20 have been amended. Claims 1-20 are pending in the office action.

Claim Rejections - 35 USC § 102

2. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

3. Claim 20 is rejected under 35 U.S.C. 102(e) as being anticipated by Aman et al., USPubN: 2004/0220947 (hereinafter Aman).

As per claim 20, Aman discloses a computer-eadable medium instructions operable by a computer, which when executed cause the computer to perform a method comprising:

instrumenting at run-time (e.g. *dynamically, ARM, API, instrumentation interfaces* - para 0126-0128, pg. 7-8; Fig. 4, Fig. 12) a hierarchical chain of parent-child transactions including a top level transition and at least one child transaction thereof (e.g. Fig. 26; para 0021-0024, pg. 2; *end-to-end, parent transaction, nested transactions* - para 0188-0195, pg 11) without modifying source codes (e.g. Fig. 4 – Note: dynamic insertion of API call invoked by the business Application – i.e. runtime - via ARM services reads on without modifying source code of transactions – see Fig. 12; see *Open Group ... invoked by the application 404 to provide the instrumentation* - para 0067, pg. 4; see *interface ... Java programming language ...*

instrumentation ... Java applications ... Java environment – para 0099-0102, pg. 6; *ARM API ... correlator is a byte* – para 0175-0176, pg. 10 – Note: byte code to implement a correlator invoked via a ARM API in Java reads on no source code being affected) associated with these transactions; and

generating correlators for each of said transactions (e.g. *correlator* -Fig. 26; para 0135-0136, pg. 8), wherein each correlator identifies said top level transaction and a parent transaction, if any, corresponding to its associated transaction (e.g. Fig. 26; para 0021-0024, pg. 2).

4. Claims 1-12 are rejected under 35 U.S.C. 102(e) as being anticipated by Labadie et al, USPubN: 2003/0195959 (hereinafter Labadie).

As per claim 1, Labadie discloses a server system method for monitoring performance of a plurality of transactions including a top level transaction and plurality of transactions relating to said top level transaction in a child parent hierarchy (e.g. Tivoli ARM ... International Business Machines - para 0005-0014, pg. 1-2; para 0036, pg. 4; *event originated; event that triggered the particular event* - para 0045-0052, pg. 4-5 – Note: ARM and Tivoli correlators reads on parent/child transaction correlators with associated measurements via API, correlation that identifying originating or triggering events/host name), comprising

for each of selected ones of said transactions, instrumenting said transaction at run-time without modifying its source code (e.g. Fig. 4A-C- Note: Middleware instrumenting of live events and transaction threads reads on without modifying corresponding web code; see EJB server, JDBC - para 0028, Fig. 1; byte stream –para 0072, pg. 6; see para. 0053-0056 pg. 5 – Note: stream of correlators in byte format being invoked at some point of transactions – see Fig.

5A, para 0059-0063, pg 5-6 – reads on byte code in Java programming environment where middleware and API invocations operate with byte format of correlators class – hence no source code involved) to obtain a performance metric corresponding thereto (e.g. para 0061, pg 5; Fig. 5A, 5B),

for each of said instrumented transactions, generating a correlator for identifying said top level transaction and a parent transaction (Note: ARM correlator reads on child/parent relationship), if any, of said transaction, and

utilizing said correlators to cross-correlate a performance metric corresponding to a parent transaction with one or more performance metrics corresponding to one or more child transactions of said parent transaction (e.g. Fig. 5B; *SOAP parameters, timestamp* – para 0073, pg. 7; Fig. 6A-C).

Labadie specifies that the server system is a EJB server with applications (para 0026, pg. 3) involving Sun Microsystems Enterprise beans; hence has disclosed that this server system is a J2EE because of the transaction-related ARM services and instrumentation on EJB Java objects (see Fig. 6A-C).

As per claim 2, Labadie discloses the step of instrumenting said transaction comprises inserting instrumentation code in a bytecode representation of said transaction (byte stream – para 0072, pg. 6).

As per claims 3-6, Labadie discloses wherein said performance metric corresponds to a response time of said transaction (*response time* - para 0005-0014, pg. 1-2); wherein said instrumentation code effects generation of a start time marker upon start of execution of said transaction and generation of a stop time marker upon completion of execution of said transaction (para 0068, pg. 6); wherein said instrumentation code generates calls to an Application Response Measurement (ARM) agent to cause generation of said stop and start time markers (service 350 – Fig. 5B; para 0005-0014, pg. 1-2) utilizing said start and stop time markers to measure a response time of said transaction (Fig. 5A, 6A).

As per claim 7, Labadie discloses generating a record for each instrumented transaction upon completion of said transaction, said record indicating said performance metric associated with said transaction (Fig. 5A-B), a parent of said transaction, and said top level transaction (Note: for each byte stream being instrumented for a ARM code instrumentation as disclosed, the top level or correlated event being monitored reads on parent or top level transaction – see para 0045-0052, pg. 4-5).

As per claim 8, Labadie discloses transmitting said transaction record to an analysis and presentation module (e.g. *PushCorrelator, GetAllCorrelator* - Fig. 6B; *Set_Context_Data, Set_Context_Info* - Fig. 6A, B; *CorrelatorTableEntry 390* – Fig. 5B).

As per claims 9-10, Labadie discloses storing of said correlators in a thread local storage stack (e.g. Fig. 4A-C - Java Virtual Machine runtime thread with Thread counter reads on thread stack in JVM, stack being inherent to a JVM runtime as evidenced by *Push*Correlator, *Pull*Correlator – Fig. 5B) in case of execution of said hierarchical transactions in a single thread (para 0037-0045, pg. 4); and storing said correlators in the stack based on a LIFO protocol (see Fig. 4A-C - Note: Java Virtual Machine runtime stack for threads recording with inclusion of associated correlators therein at runtime, reads on LIFO protocol of a given stack).

As per claim 11, Labadie discloses removing a correlator from said stack upon completion of a transaction associated with said correlator (*PullCorrelator* – Fig. 6B).

As per claim 12, Labadie discloses wherein said top-level transaction is initiated in response to a request received from a web server (e.g. Init_Correlator – Fig. 6A- Note: any partner event in the flow of threads – see Fig. 4A-C- reads on a top thread leading to other thread downstream based on the first request and Init – see *InitClientMWare* –Fig. 6A, in light para 0034, pg. 4).

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claim 13, 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Labadie et al, USPubN: 2003/0195959 (hereinafter Labadie), and further in view of Hind et al, USPN: 7,003,565 (hereinafter Hind).

As per claims 13-14, Labadie does not explicitly disclose wherein said web server transmits a cookie to said application server together with said request; and further utilizing said cookie to generate said top level correlator. But the client state being collected and passed (see Fig. 5A-B) over to different servers (plug-in middleware, DCS correlator service) using the instrumentation service (ARM) to record correlator as shown by Labadie (see para 0028-0032, 0034-0036) entail client runtime data/events to be passed from services to services to enable correlation of thread or partners being enumerated for a process request or analysis thereof(see Fig. 4A-C). The use of cookie at a given machine to store client data for repeated usage – so to obviate creation of addtitional discovery resources -- was concept used in the data collecting paradigm by Hind so that by using these record or cookie under the provision of message as to communicate with servers (see Fig. 3A; correlator – col. 8, lines 20-67) Hind's collection of correlator-type of data can support service as to improve QoS delivery or administrative policy enforcing. Hence, in the same endeavor as analyzing performance of transaction as Labadie, Hind provides in-bound and out-bound message with cookie data so to provide very specific client data for server to enforce quality control transaction using correlation information therein (see Fig. 6) thus to alleviate dependency of information interchanges from many sources of data providers (or linked servers) as cookie messaging can yield latest state of client information. Hence, in view to the multiple agent messaging as required in Labadie's correlator record passing, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide cookie messaging by Hind, i.e. using said cookie data to implement or to support correlator data collection as purported by Labadie. One skill in the art would be motivated to do so because cookie data from one sending edge server to the next would alleviate extraneous discovery resources for these data reflect the most accurate and dynamic state of the client information being passed (see Hind, col. 16, bottom to col 17 line 34) such that by utilizing this cookie approach, the servers can make use of the most up-to-date state of a client/requesting source data to fulfill the quality of transaction as approached by Labadie's instrumentation service, or to facilitate the enforcement of transaction security as endeavored by Hind's Qos paradigm.

7. Claim 15-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Labadie et al, USPubN: 2003/0195959 (hereinafter Labadie), and further in view of Bansal et al., USPubN: 2003/0120593 (hereinafter Bansal)

As per claim 15, Labadie discloses a method for monitoring performance of at least two Java transactions executing in two separate processes and being related to one another as parent-child transactions (re claim 1), comprising:

instrumenting each of said transactions at run-time by modifying its respective bytecode representation (e.g. *SOAP, jvmID* – para 0036-0044, pg. 4; JMS, para 0049, pg. 4; *toString* – para 0053, pg. 5; Fig. 3, 5, 6 – Note: threads in a java-based event capturing SOAP, using instances of correlator's methods – e.g. toString -- and JVM runtime entails execution of code by modifying a runtime JVM bytecodes; *byte stream* –para 0072, pg. 6 – Note: stream of correlators in byte format being invoked at some point of transactions – see Fig. 5A, para 0059-0063, pg 5-6 – reads on bytecode in Java programming environment where middleware and API invocations operate with byte format of Java classes or interfaces) to obtain a selected performance metric corresponding thereto, generating a correlator corresponding to said top-level transaction.

Labadie discloses utilizing RMI (see ORB, para 0028, pg. 3) to send said top-level correlator incorporated in a header of an IIOP message to said child transaction, and generator another correlator corresponding to said child transaction (Fig. 5A-B; *header* - para 0064-0066, pg. 6); but does not explicitly teach RMI over IIOP. The use of message over IIOP in a J2EE based network is taught by Bansal (Fig. 23) who also teaches using of ARM to instrument and measure application data for performance reporting (see para 0922-0969, pg. 38-40). Since Labadie is also suggesting performance analysis in a similar context where message containing correlator data are passed in a interoperability Enterprise Java network (para 0026, pg. 3), it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide a layer of IIOP as in Bansal's message passing above among the ORB layer pertinent to this J2EE paradigm in order for Labadie's RMI invocation (over ORB) or correlator record passing to benefit of the core service of the ORB based on IIOP as heterogeneous format data can be reconverted from one format into another to fulfill the path of the data being transferred in this enterprise communications means.

As per claims 16-19, these claims include the subject matter of claims 3-5 or 6; hence will incorporate the corresponding rejection as set forth therein, respectively.

Response to Arguments

8. Applicant's arguments filed 4/9/07 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

35 USC § 102 Rejection:

(A) Applicants have submitted that according to IEEE dictionary definition, the term instrumenting means inserting software into a transaction for monitoring a performance metric (Appl. Rmrks, pg. 8 top) and therefore, Labadie as provided via Figure 4 does not teach instrumenting a transaction at run-time (Appl. Rmrks pg. 8, middle) without modifying a source code. The rejection has set forth cited portions that point to an instrumentation of runtime transactions using ARM service by which APIs or plug-in are invoked to and from middleware components in order to create observation points into stream of data, e.g. using methods of correlator in Java byte format (see para. 0053-0056 pg. 5; byte stream – para 0072 in light of Fig. 5A-B correlators classes/interface method) so to effectuate instrumentation as to count for a

Process a thread sequence, for example. There is no mention about source code anywhere in the instrumentation method by Labadie, according to whose approach, only the byte form of the correlator Java are invoked and inserted at some points of the transaction stream based on some provider's plug-in action in the communication paradigm of Labadie's DCS working with EJB server and ORB midleware and SOAP messaging. To the extent of the IEEE definition as set forth above, instrumentation (of a transaction) by Labadie is perceived as insertion of software invocations during transaction processes to provide means to monitor counts – see para 0053-0057, pg. 5; Fig. 6A,B,C and their *call sequence* of correlator classes – see para 0073, pg. 6). Modifying of source code is not even remotely mentioned in the above dynamic insertion of call invocations performed by Labadie's ARM. ARM was – at the time the invention was made - known to be called upon during runtime of business transactions and in Labadie's EJB-based environment, Java bytecodes would be predominantly the form standard for compiler in such applications to use and create interface class, and this is shown by the specifications of a SOAP whereby a JVM of a host is to be identified (see para 0036-0045, pg. 4) and the classes intended for the correlator according to the call sequences as shown above. If the Invention were to provide support (USC § 112 requirements) for the recited 'without modifying its source code' in light of the above IEEE definition, this limitation is seen or being described as modifying bytecodes (see Specifications, pg. 18; Fig. 8C) in a context of using ARM API and plug-ins. In this light, the instrumentation by the claimed invention and that of Labadie appear to be surprisingly almost identical. But the *Specifications* is not necessarily read into the claim as the claim is interpreted by one of ordinary skill at the time of prosecution. Labadie teaches effectuating of plug-ins to trigger correlator (byte form) creation; setting Java methods of the correlators in view to recording counts of process within a particular set of threads or points thereof. Since byte code is a common form in Java/JVM based application runtime for invoking of Java class at runtime, the above teaching is construed as instrumenting at runtime by inserting software into byte code -- without having to create the very high-level source code and recompiling; i.e. Java bytecodes being a intermediate form between source code and binary executable. Thus, modifying bytecodes for instrumentation by Labadie has met the 'without modifying source code' limitation. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

35 USC § 103(a) Rejection:

(B) Applicants have submitted that claim 15 recites 'its respective bytecode representation' and that the SOAP parameter as proffered by the Office Action to convert a string data is not what is claimed; therefore Labadie's as presented in the rejection does not cure the *bytecode* of claim 15 in the same line as the failure to fulfill the 'instrumenting ... at runtime' of claim 1 (Appl. Rmrks pg. 8, bottom, pg 9, top). Section A above has set forth the Java environment by Labadie and the byte format of stream upon which Java-implemented correlator class and methods are dynamically created using a SOAP type of data carrier, the specifications by which SOAP extensible messaging lead to the creation of methods and specification of points (of the transaction threads) at which process of a given ID can be monitored (see related text of Figures 4, 5, 6). Labadie has disclosed byte code format and instrumentation using plug-in and ARM API and invocations of runtime JVM-based correlators methods whose class/interface are customized (according to specifications messages) to perform the monitoring of transactions streams. If Applicants are not convinced that the concept of Java-based correlators in light of EJB/ARM driven applications does not necessarily mean that bytecode is virtually the predominant form of Java code to instrument, Applicants however, fail to point out which part of Labadie is clearly teaching that Java classes of the correlators are dynamically supplied as source code (here: *source code* means a higher level than bytecodes) in the ARM paradigm, such source code then modified with source code constructs insertions, then recompiled into a instrumented class as an executable binary. The *bytecodes* argument is therefore non-persuasive. The argument against Bansal is not specifically responsive to the specifics and mechanics of the rationale set forth for the 103(a) rejection; hence is treated as a mere *non-prima facie* assertion without weight. Java-based enterprise networks with Open Source ARM for code instrumentation was a known concept at the time the invention was made, and bytecode manipulation in this Open Source system is exemplified in Aman; i.e. source code level as the conventional high-level source format not implicated in this ARM based environment; and many business enterprises (e.g. see Berent et al, Turner et al. - hereinafter) applying Java in the ARM methodology can corroborate that 'bytecode' has been the predominant format to support this runtime JAVA code modification and/or correlator-based transaction event/stream monitoring. Labadie happens to be one such environment where correlators in Java form are dynamically inserted and invoked to effect this monitoring, i.e. only bytecodes as an easily-maneuverable source format required for the JVM or JIT runtime.

(C) Applicants have submitted that Aman's disclosing of dynamic library API and the erroneous conception of 'instrumenting' by the Examiner are such that Aman fails to teach or suggest the claimed limitation of claim 20 (Appl. Rmrks pg. 9, bottom). In reply, the argument falls under the ambit of the argument being previously raised against the rejection of claim 1; and is thus referred to section A for what this 'instrumenting' represents in light of Labadie's approach. In short, the argument is not persuasive.

The claims will stand rejected as set forth in the Office Action.

Conclusion

9. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Berent et al, USPubN: 2004/0019886; teaches ARM in a Java based runtime compilation with byte-codes instrumentation triggered via plug-ins and profiling of business process.

Turner et al, "Application Response Measurement of Distributed Web Services" also teaches bytecode instrumentation to measure e-business-type requests.

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

[illegible]

Tuan A Vu
Patent Examiner,
Art Unit 2193
April 15, 2007